Design Patterns for Model Transformations: Current research and future directions

K. Lano[1], S. Yassipour-Tehrani[1]

[1]Dept of Informatics, King's College London, Strand, London, UK

Abstract. There is increasing interest in the use of design patterns for model transformations, and a number of such patterns have been proposed. In this paper we survey previous work on transformation design patterns, and identify priorities for future research.

1 Introduction

Design patterns have proved to be an effective concept to improve the quality of software systems, with classic patterns such as Iterator, Facade and MVC now part of the standard vocabulary of software developers and provided as essential elements of programming languages and of software development environments. Model transformation (MT) development should also be able to benefit from the standardised solutions and expertise embedded in design patterns: currently such development is often carried out in an ad-hoc manner because of the novel nature of the languages and processing involved.

Transformation development requires new or adapted specification and design patterns because of the specialised features of MT programming and languages:

- Transformations involve complex structured data (models) and navigation through and manipulation of such data.
- Transformations involve unconventional processing such as graph rewriting and partially-specified execution orders.
- Hybrid MT languages involve a mix of declarative and imperative language elements.
- Languages supporting bidirectional transformations involve complex processing to synchronise or transform models in either source to target or target to source directions.

2 Research into MT design patterns

The first works on MT design patterns identified language-specific patterns for the ATL [8] and QVT-R [18] languages:

ATL: In [2, 6] ATL-specific patterns were identified: Transformation Parameters; Multiple Matching; Object Indexing; Many-to-one; Many-to-many; Custom Tracing.

QVT-R: In [12] QVT-R patterns such as Enable conditions in only one Direction, and Simulation of Key were introduced. In [9] the use of marker relations in model copying transformations in QVT-R was detailed.

In some cases, such patterns simply gave techniques for circumventing limitations of the specific MT languages.

Subsequently, work was carried out to identify language-independent MT design patterns, based on generalisations of the language-specific patterns or adaptions of classical design patterns for MT: in [1, 6, 4, 5, 10, 11, 13, 14, 16] such patterns were described. These included Auxiliary Metamodel, a generalisation of the ATL pattern Transformation Parameters, and Map Objects Before Links, a general strategy for model copying and migration transformations.

In [16] the patterns were grouped into five categories:

1. **Rule modularisation patterns:** Phased Construction; Structure Preservation; Entity Splitting/ Structure Elaboration; Entity Merging/Structure Abstraction; Map Objects Before Links; Parallel Composition/Sequential Composition; Auxiliary Metamodel; Construction and Cleanup; Recursive Descent; Replace Explicit Calls by Implicit Calls; Introduce Rule Inheritance.
2. **Optimisation patterns:** Unique Instantiation; Object Indexing; Omit Negative Application Conditions; Replace Fixed-point by Bounded Iteration; Decompose Complex Navigations; Restrict Input Ranges; Remove Duplicated Expression Evaluations; Implicit Copy.
3. **Model-to-text patterns:** Model Visitor; Text Templates; Replace Abstract by Concrete Syntax.
4. **Expressiveness patterns:** Simulating Multiple Matching; Simulating Universal Quantification; Simulating Explicit Rule Scheduling.
5. **Architectural patterns:** Phased Model Construction; Target Model Splitting; Model Merging; Auxiliary Models; Filter Before Processing.

Patterns for specific categories of transformation have also been identified:

Bidirectional transformation patterns: Auxiliary Correspondence Model; Cleanup before Construct; Unique Instantiation; Phased Construction for bx; Entity Merging/Splitting for bx; Map Objects Before Links for bx [17].

Data migration patterns: Migrate along Domain Partitions; Measure Migration Quality; Data Cleansing [19].

3 Future research directions and challenges

Because of the novelty of the MT field, there is as yet insufficient evidence for the effectiveness of most of the identified patterns. Work is needed to survey applications of MT patterns in practice and to evaluate their utility. Work is also needed to identify appropriate classifications for MT patterns, to identify connections between patterns, and useful compositions of patterns. Techniques

and guidelines for the selection of MT patterns need to be developed and incorporated into MT engineering environments. Patterns for new types of transformation, such as transformations at runtime or streaming transformations, are also needed. Finally, research into transformation anti-patterns and techniques for the identification of 'bad smells' in transformation specifications is needed.

References

1. A. Agrawal, A. Vizhanyo, Z. Kalmar, F. Shi, A. Narayanan, G. Karsai, *Reusable Idioms and Patterns in Graph Transformation Languages*, Electronic notes in Theoretical Computer Science, pp. 181–192, 2005.
2. ATL Design Patterns, https://wiki.eclipse.org/ATL/Design_Patterns, 2015.
3. I. Bayley, H. Zhu, *On the composition of design patterns*, QSIC 2008, IEEE Computer Society, 2008, pp. 27–36.
4. J. S. Cuadrado, F. Jouault, J. G. Molina, J. Bezivin, *Optimization patterns for OCL-based model transformations*, MODELS 2008, vol. 5421 LNCS, Springer-Verlag, pp. 273–284, 2008.
5. K. Duddy, A. Gerber, M. Lawley, K. Raymond, J. Steel, *Model transformation: a declarative, reusable pattern approach*, 7th International Enterprise Distributed Object Computing Conference (EDOC '03), 2003, pp. 174–185.
6. J. Bezivin, F. Jouault, J. Palies, *Towards Model Transformation Design Patterns*, 1st European Workshop on Model Transformations, 2005.
7. V. Bollati, J. Vara, A. Jimenez, E. Marcos, *Applying MDE to the (semi-)automatic development of model transformations*, Information and Software Technology, 2013.
8. Eclipsepedia, *ATL User Guide*, http://wiki.eclipse.org/ATL/ User_Guide_-_The_ATL_Language, 2014.
9. T. Goldschmidt, G. Wachsmuth, *Refinement transformation support for QVT relational transformations*, FZI, Karlsruhe, Germany, 2011.
10. M. E. Iacob, M. W. A. Steen, L. Heerink, *Reusable model transformation patterns*, Enterprise Distributed Object Computing Conference Workshops, 2008, pp. 1–10, doi:10.1109/EDOCW.2008.51.
11. J. Johannes, S. Zschaler, M. Fernandez, A. Castillo, D. Kolovos, R. Paige, *Abstracting complex languages through transformation and composition*, MODELS 2009, LNCS 5795, pp. 546–550, 2009.
12. J. Kiegeland, H. Eichler, *Medini-QVT*, http://projects.ikv.de/qvt, 2014.
13. K. Lano, S. Kolahdouz-Rahimi, *Model transformation design patterns*, ICSEA 2011, IARIA, 2011, pp. 263–268.
14. K. Lano, S. Kolahdouz-Rahimi, *Optimising Model-transformations using Design Patterns*, MODELSWARD 2013.
15. K. Lano, *The UML-RSDS Manual*, www.dcs.kcl.ac.uk/staff/kcl/uml2web/umlrsds.pdf, 2014.
16. K. Lano, S. Kolahdouz-Rahimi, *Model Transformation Design Patterns*, IEEE Transactions in Software Engineering, 2014.
17. K. Lano, S. Kolahdouz-Rahimi, *Model transformation design patterns for bidirectionality*, FSEN 2015.
18. OMG, *MOF 2.0 Query/View/Transformation Specification v1.1*, 2011.
19. M. Wagner, T. Wellhausen, *Patterns for data migration projects*, www.tngtech.com, 2011.